Data Grids and Data Grid Performance Issues

Brian L. Tierney (bltierney@lbl.gov)
Data Intensive Distributed Computing Group
Lawrence Berkeley National Laboratory
(currently on leave at CERN, IT/PDP/TE)

Outline

- Part I
  - — Introduction
  - — Data Grid Architecture
  - — Existing Data Grid Systems
    - DPSS and SRB
- Part II
  - — Current Data Grid Work
    - Globus + LBNL work
    - other data grid projects
- Part III
  - — Data Grid Performance Issues
    - network and TCP Issues
    - application design considerations

Introduction

Problem

- The Problem
  - — To enable a geographically distributed community to performance analyses on petabytes of data efficiently and cost-effectively.
- The Proposed Solution
  - — The Data Grid

Computational/Data Grids

- Grid / Computational Grid:
  - — The integration of various approaches used for coupling geographically dispersed resources
  - — analogy with the grid that supplies ubiquitous access to electric power
  - — Basic grid services are those that locate, allocate, coordinate, utilize these resources
- Data Grid:
  - — services for handling remote access to large data sets in a grid environment

Grids

- Q: Are Grids something new? A: Not really.
- The concept of using multiple distributed resources to cooperatively work on a single application is not new.
  - — 1979-81: “networked operating systems”
  - — 1988-91: “distributed operating systems”
  - — 1993-94: “heterogeneous computing”
  - — 1995-96: “parallel distributed computing” (e.g.: pvm)
  - — 1996-98: “metacomputing”
  - — 1998: The **Grid**
  - — 1989-99: remote data visualization
  - — 1995-99: “data intensive distributed computing”
  - — 1999: The **Data Grid**

Why are Grids now a "hot topic"?

- New idea:
  - — much better security / authentication model
    - user: single login for a large collection of resources at multiple sites
    - sites: control over what users/groups have access to resources
  - — PKI-based
- Networks finally getting fast enough
  - — your WAN connection can now be even faster than your LAN!
- The ideas from "parallel distributed computing" papers of the mid-90's now become very exciting

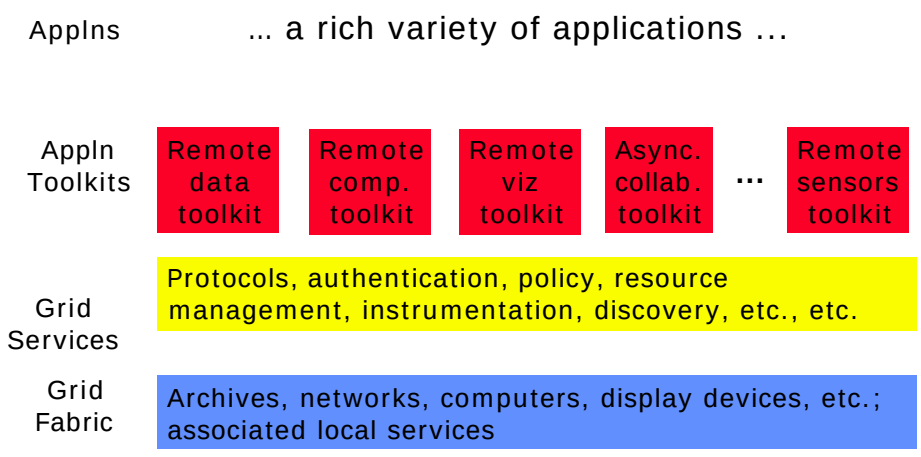
Grid Services

- Grid services include:
  - — authentication
  - — resource location
  - — resource allocation
  - — configuration
  - — communication
  - — remote file access
  - — fault detection
  - — executable management
- You should learn much more about these from Prof. Aloisio tomorrow

Data Grid Services

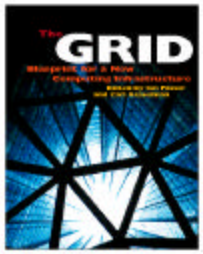
- The term “Data Grid” describes services that are aimed at data intensive grid applications. These services include:
  - — data migration tools that are optimized for transferring large data sets over WANs
  - — data set discovery and replication tools
    - global name space for data archived at multiple sites
  - — data caches / cache management services
  - — replica management services
  - — replica selection services

An Integrated Grid Architecture



Grid Software

- Globus
  - — From Argonne National Lab and USC/ISI
  - — http://www.globus.org/
  - — editors of the “Grid book”



- Legion
  - — University of Virginia
  - — http://www.cs.virginia.edu/~legion/
- Other speakers will tell you much more about Globus next week

Why are Data Grids important?

- Researchers often are not at the same location as the data source
- Compute cycles are often not at the same location as the data source or the data archive
- Scientific data sets are HUGE!
  - — CERN LHC particle physics detector will soon produce over 4 TB of data per day
  - — A global warming simulation on a supercomputer can generate a TB of data
  - — NASA EOS Satellites will soon generate > 2 TB per day

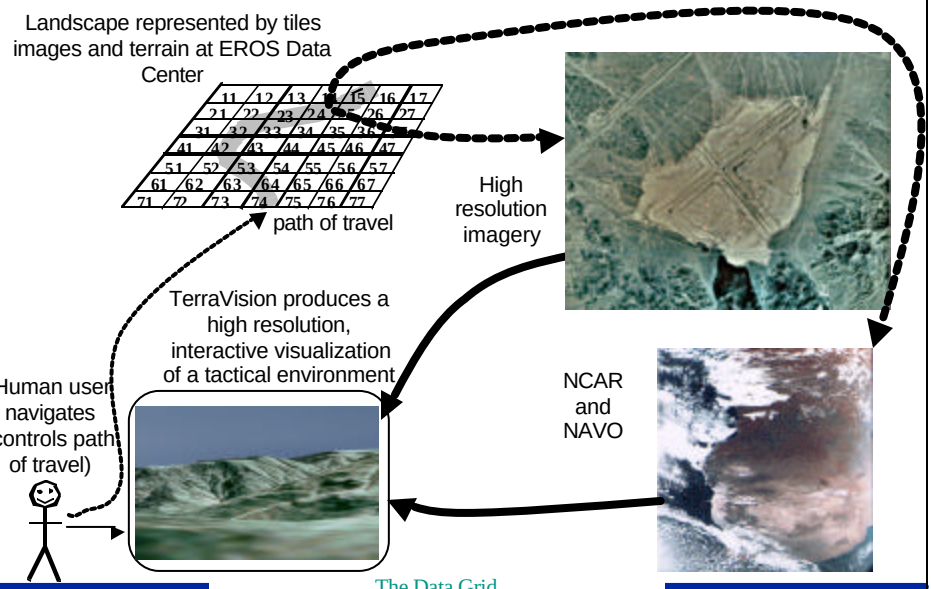
Data Grid Applications

- Many scientific fields have these characteristics:
  - a large, geographically dispersed group of researchers
  - require access to huge amounts of data
- These fields include:
  - combustion modeling
  - climate modeling
  - MEMS device testing and fabrication
  - High Energy Physics
  - Oceanography
  - Astronomy
  - many others

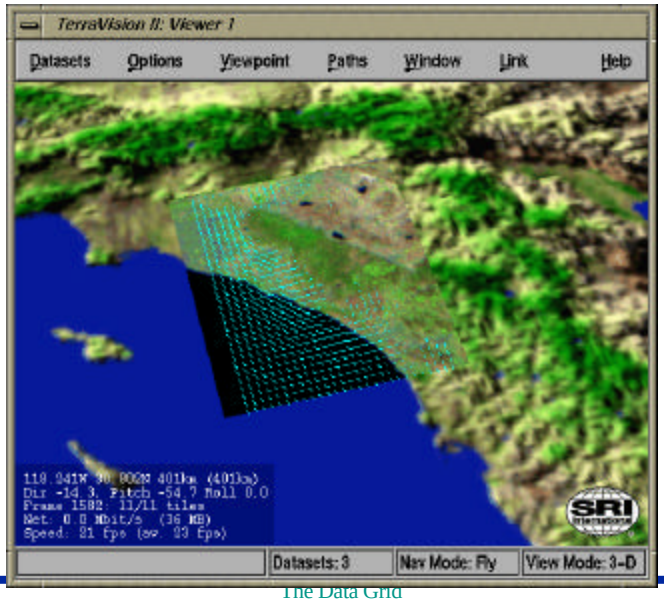
Background

- My group at LBNL has been involved with Data Grid-like applications for several years.
- Much of our experience came from the DARPA-sponsored “Magic Project” (http://www.magic.net)
  - 1993-98
  - one of the “gigabit network testbed” projects
  - Goal: high-speed access to remote terrain image databases

TerraVision, DPSS, and MAGIC Testbed Provides Real-time Visualization of Aggregated Data

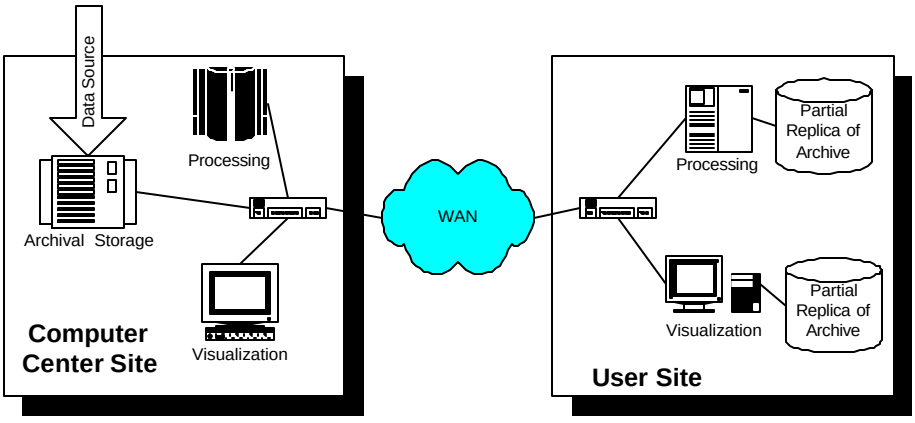


TerraVision Image of 3D Terrain with Wind Vectors

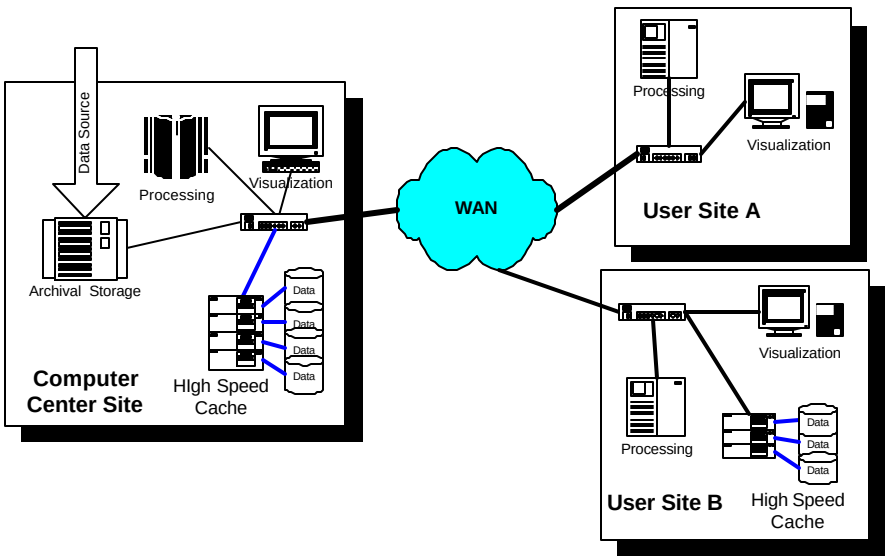


Data Grid Architecture

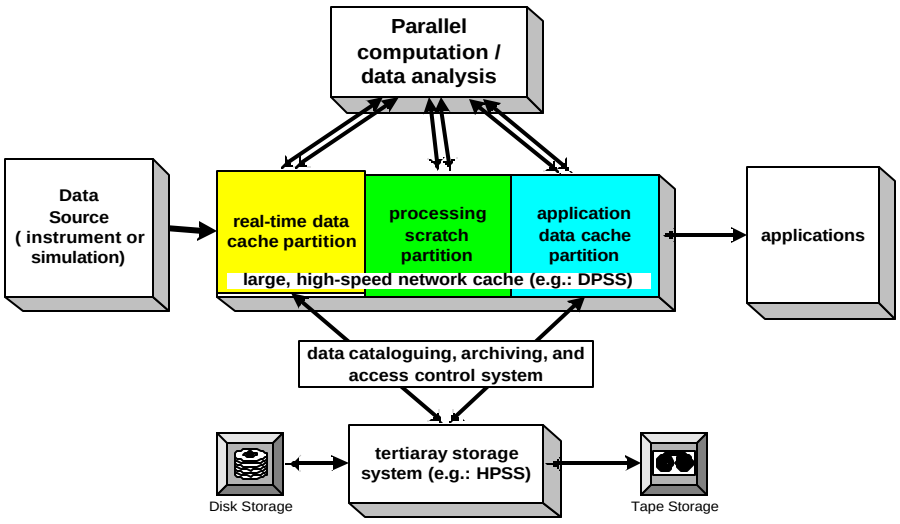
Remote Access to a Large Data Archive: Traditional Method



Remote Access to a Large Data Archive using a High-Speed WAN and a Data Cache



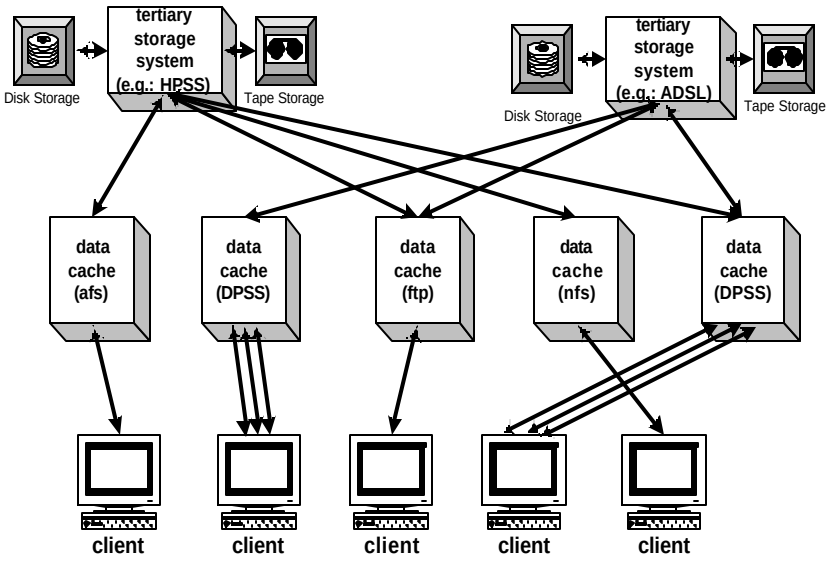
Cache-based Data Grid Architecture



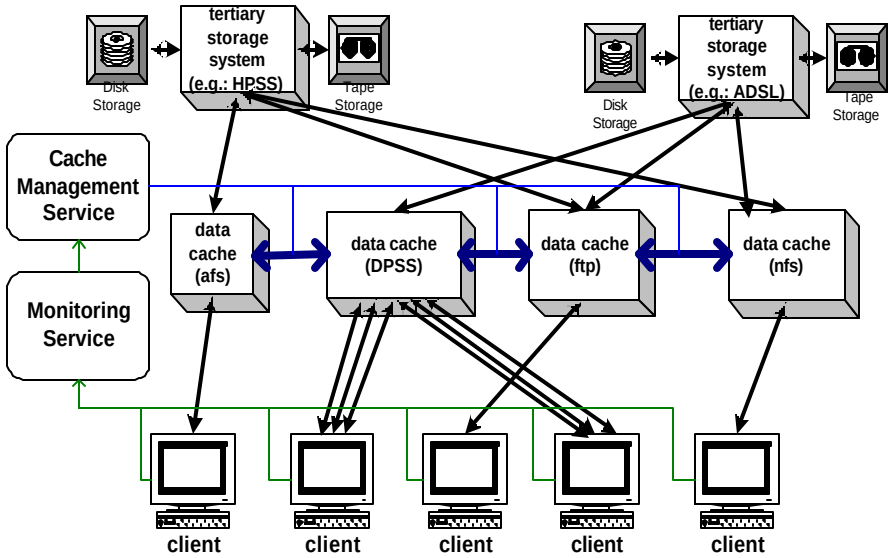
Key features of the architecture

- Allows for high-speed access to very large scientific data sets using an http-like model
  - — users of the web don't download an entire web site, only the parts required immediately
- Enables remote scientific visualization applications, remote data analysis, etc., in a similar manner
  - — no longer need to copy entire data set locally
- Key Components
  - — high-speed data cache tuned for WAN access
  - — high-speed tertiary storage interface
  - — data cataloguing system
  - — access control system

The Data Aspects of the Grid: Current State



The Data Aspects of the Grid: Future State



Data Grid Services

- Resource reservation and co-allocation mechanisms
  - — for storage systems, data caches, networks, and compute hosts
- Data filtering services
  - — often more efficient to do some filtering at the data archive site
- Metadata Services
  - — application metadata
    - NetCDF, HDF, XML
  - — replica metadata
  - — system configuration metadata
    - database of storage system characteristics

Data Grid Security Issues

- Data Grids must:
  - — utilize resources from multiple administrative domains
  - — respect local policies on who can use what, and how much, how long, etc.
  - — enforce access control on data

Typical Client Use of Data Grid Services

A client might go through the following steps:

1) use authentication system to “log on” to the Grid
2) use global file name to find current location of data set and all replicas
3) locate any data caches that are well connected to client host, and reserve cache space
4) if necessary, copy the data set from the archive to the data cache
5) query network advice service for recommended network parameters (QoS, TCP options, etc.)
6) make network reservation with a bandwidth broker

Data Cache Research Topics

Issues for using Data Caches scattered across the Internet:

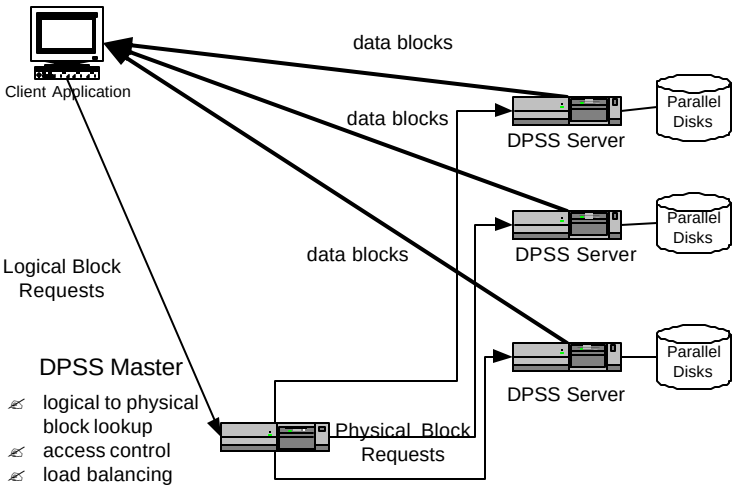
- — how and when to migrate files
- — how to determine which data cache should be used
- — when is it better to move the processing to the data, instead of the data to the processing
- — how to reserve space on the data cache
- — how to achieve high data rates across wide area networks
- — how to provide a global data set name space
- — how to ensure data set consistency
  - for the most part we have been assuming read-only data sets

Existing Data Grid Systems

The Distributed Parallel Storage Server (DPSS)

- At LBNL we have implemented a network data cache called the DPSS
  - — provides high-speed parallel access to remote data
  - — Similar to a striped RAID system, but tuned for WAN access
    - data is striped across both disks and servers
  - — On a high-speed network, can actually access remote data faster that from a single local disk
    - 71 MB/sec vs. 10 - 20 MB/sec

DPSS Architecture



Typical DPSS implementation

- 4 UNIX workstations (e.g. Pentium 400+ with Linux), each with:
  - — 4 - 6 disks
  - — a high-speed network interface (e.g.: 1000BT)
- This configuration can deliver an aggregated data stream of about 1 Gbit/s (125 MBytes/s) using these relatively low-cost, “off the shelf” components by exploiting the parallelism of:
  - — four hosts
  - — twenty disks
  - — four network interfaces

1 TeraByte High Performance Data Cache < $15K (USD)

- Example: a 4 server DPSS system
- Sample DPSS Server Configuration:
  - — Linux / Pentium (700 MHz PIII) server: $1100
  - — 3ware storage controller: $300 (essential component!)
  - — Netgear 1000BT card: $310
  - — 45 GB Western Digital IDE Disks; 6 per server: $260/each
- Total Capacity: 24 disks x 45 GB = 1.08 TeraBytes
- Total Throughput: 250 Mbits x 4 servers = 1 Gbps
- Total Cost: $3300 x 4 servers = $13.2K
- Note that cost is no longer dominated by storage cost, as was the case only 2 years ago!

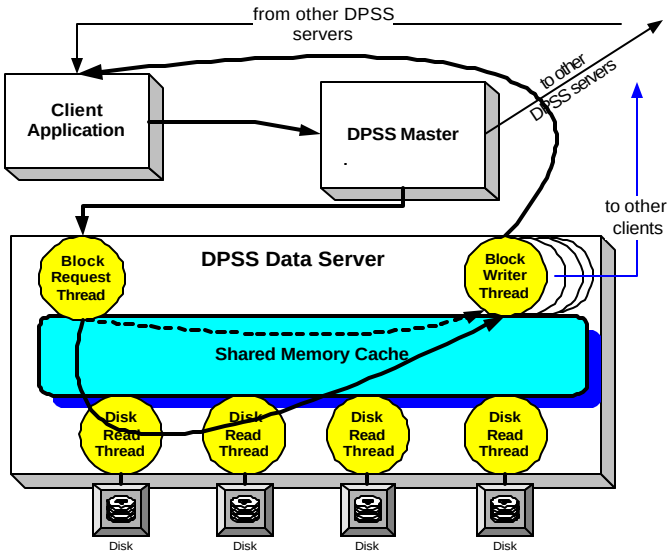
DPSS Features

- The DPSS supports:
  - — Maximum possible WAN throughput
    - parallel transfers (from multiple servers and multiple sockets per server)
    - Network tuning: easily tunable TCP buffers
    - (more on this tomorrow)
  - — Client API supports most all UNIX I/O functionality
    - threaded client library for optimal performance
      - — especially on SMP hosts
  - — replicated data for fault tolerance
  - — works on Linux, Solaris, IRIX, and FreeBSD
  - — Note: DPSS is designed to be used as a cache, not a storage system: no way to back it up

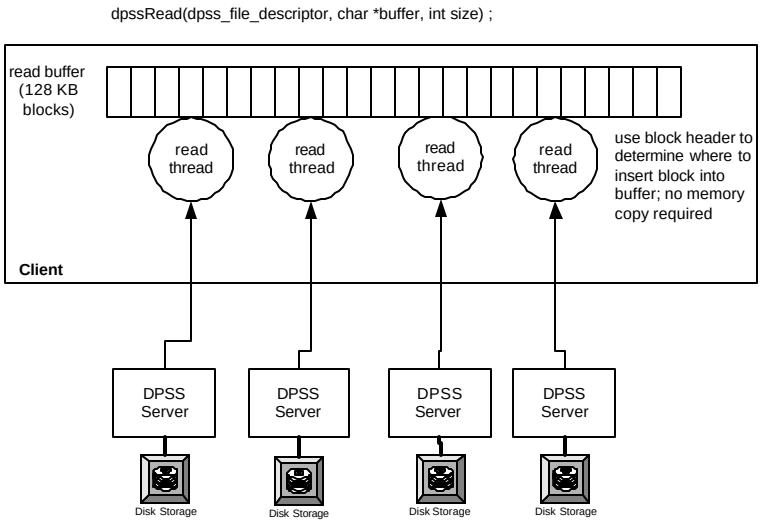
DPSS client API

- Modeled on Unix I/O
- C library with the following routines:
  - — dpssOpen("x-dpss://hostname/setname",mode)
  - — dpssAlloc()
  - — dpssRead()
  - — dpssWrite()
  - — dpssLseek()
  - — dpssClose()
- Read/Write calls have a thread per DPSS server
  - — client scales with number of servers

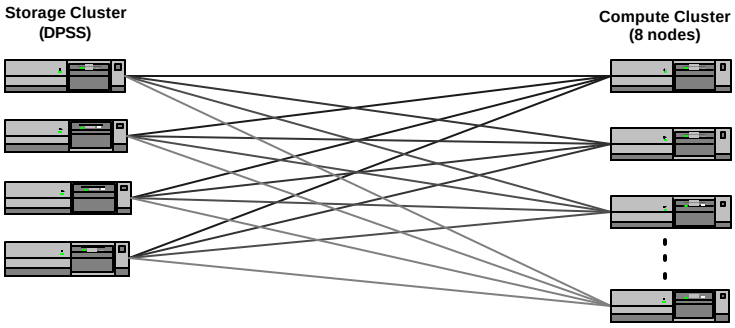
DPSS Server Architecture



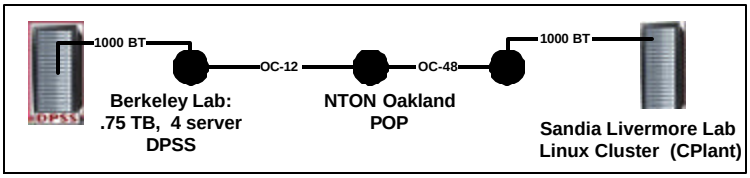
Threaded Read: dpssRead()



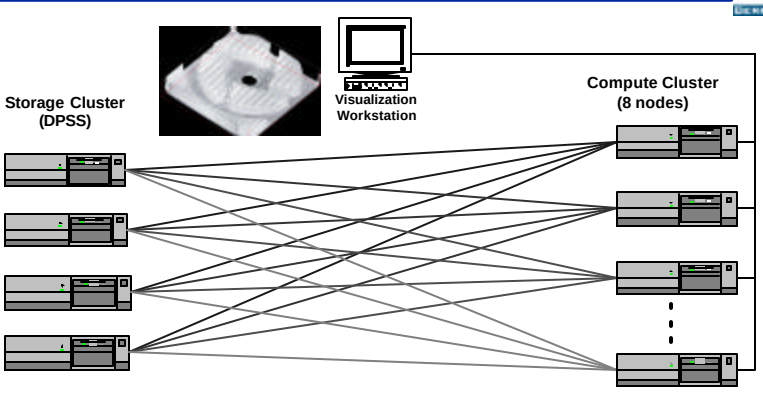
DPSS Performance

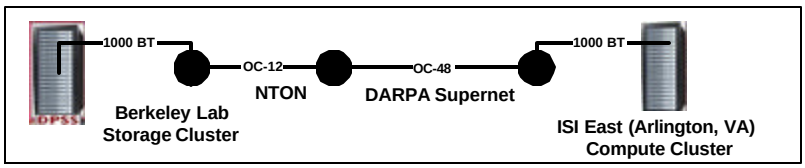


Total Throughput (single dataset to a single cluster application):
570 Mbits/sec (71 MB/sec) on 32 data streams (17 Mbits/sec/stream)



Sample DPSS application: MEMS Data Processing / Visualization

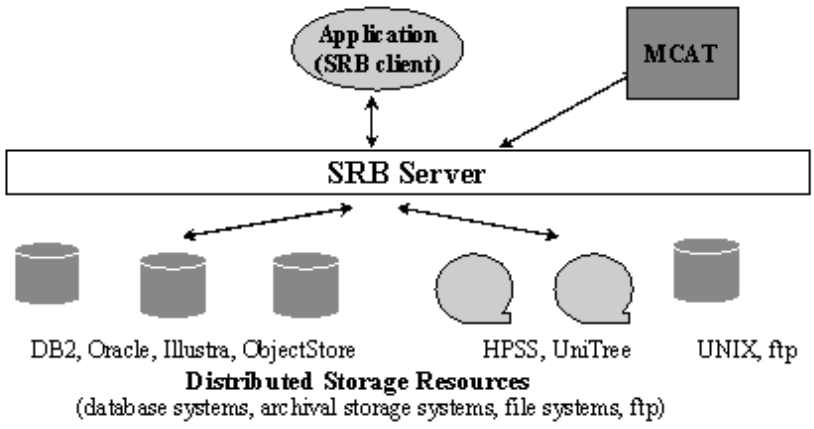




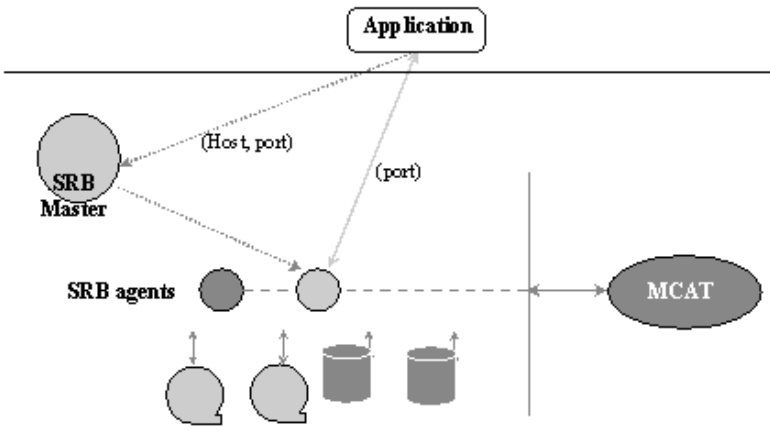
The Storage Resource Broker (SRB)

- Developed at the San Diego Supercomputer Center (SDSC)
  - Available from http://www.npaci.edu/DICE/SRB/
- SRB provides distributed clients with uniform access to diverse storage resources, including:
  - UNIX file system
  - archival storage systems such as UNITREE and HPSS
  - database Objects managed by various DBMS including DB2, Oracle and Illustra
- SRB Client library provides an API for communicating with SRB Servers
- MCAT (Metadata Catalogue):
  - SRB metadata is managed by an MCAT server
    - based on relational database technology
  - stores metadata associated with data sets, users and resources
    - includes the information on access control and "collections"

SRB



The SRB Process Model



SRB Data Model

- Logical data sets
  - — Each data set stored in SRB has a logical name, which may be used as a handle for data operation
  - — MCAT keeps track of the physical location of the data sets
  - — Data sets belonging to the same collection may physically reside in different storage systems
- Collections
  - — Similar to file system paradigm, data sets can be arranged in a directory-like structure, called a “collection”
    - Collections can be physically distributed data sets and/or sub-collections

SRB Data Model

- Containers
  - — Archival storage systems such as HPSS are not suitable for storing a large number of small files
    - container concept created to address this problem
  - — many small files can be aggregated in the cache system before storage in the archival storage system

SRB Features

- UNIX like utilities (e.g., ls, cp, chmod, etc) for manipulating and querying datasets
- A JAVA based SRB client GUI: performs a variety of client level operations, like replication, copy and paste, metadata query, etc.
- Authentication
  - — supports Globus GSI, SEA (developed at SDSC) and plain text passwords
- Tickets - a mechanism for sharing data among users
  - — The owner of a data object or collection may grant read only access to user through issuance of a ticket (a 10 character alphanumeric string)
  - — A ticket can be issued to either MCAT registered or unregistered users
- Data replication:
  - — Provides automatic creation of replica during data object creation
- Platforms supported:
  - — AIX, Solaris, DEC OSF, SGI and the Cray C90
  - — The MCAT catalog runs on both the Oracle and DB2 DBMS

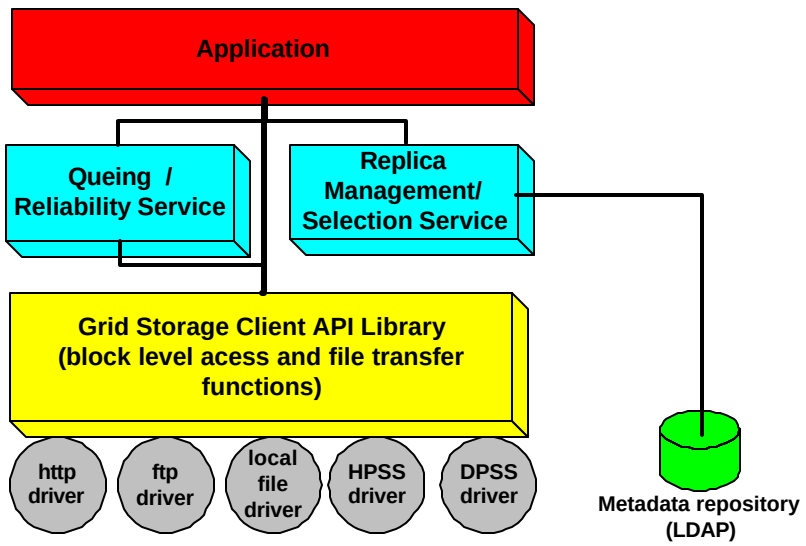
DPSS + SRB

- DPSS support has recently been added to SRB
  - — DPSS can now be used as a SRB device

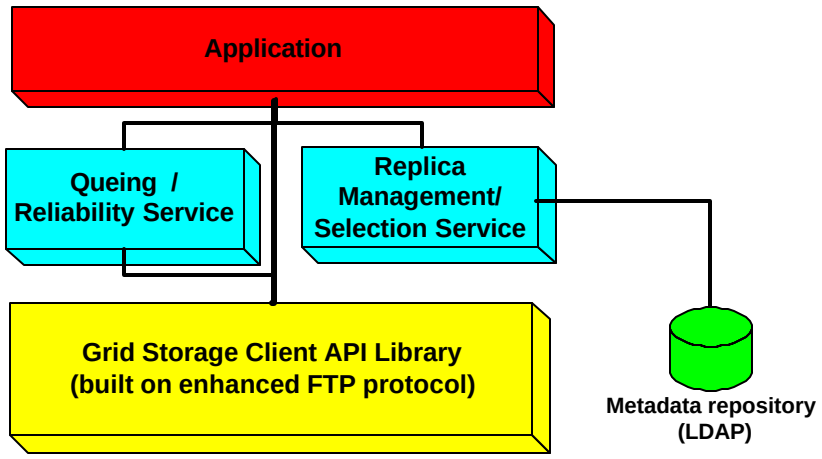
End of Part I

Current Data Grid Work (Globus + LBNL)

Globus Data Grid Architecture: version 1



Globus Data Grid Architecture: version 2



Globus Data Grid effort: ftp protocol-based remote I/O library

- ANL and LBNL are developing a high-performance transfer library based on the ftp protocol (with some extensions)
  - Built on Globus-IO (uses GSI: Globus PKI-based security)
  - Supports parallel transfers (from multiple servers and multiple sockets per server)
  - Network tuning: auto-tuned TCP buffers, etc.
  - Client API supports most all UNIX I/O functionality, including:
    - full and partial file read/write
    - asynchronous reads/writes
- See: http://www.globus.org/datagrid/

Motivation for a Common Data Access Protocol

- Existing distributed data storage systems
  - — DPSS, HPSS: focus on high-performance access, utilize parallel data transfer, striping
  - — DFS: focus on high-volume usage, dataset replication, local caching
  - — SRB: connects heterogeneous data collections, uniform client interface, metadata queries
- Problems
  - — Incompatible protocols
    - Each require custom client
    - Partitions available data sets and storage devices
  - — Each protocol has subset of desired functionality

The slide from Steve Tuecke, ANL

Common Data Access Protocol and Storage Resource Managers

- Solution: Use a common, *extensible* transfer protocol
- Grid encompasses “dumb” & “smart” storage
- support all base functionality
  - — “Put” and “get” as essential mechanisms
  - — Integrated security mechanisms, of course
- Storage Resource Managers can enhance functionality of selected storage systems
  - — E.g., monitoring progress, reservation, queuing, striping
  - — Plays a role exactly analogous to “Compute Resource Manager”
- Common protocol means all can interoperate

The slide from Steve Tuecke, ANL

A Universal Access/Transport Protocol

- The protocol must allow us to build a suite of communication libraries and related tools that support:
  - — GSI security
  - — Third-party transfers
  - — Parameter set/negotiate
  - — Partial file access
  - — Reliability/restart
  - — *Logging/audit trail*
  - – Integrated instrumentation
  - – Parallel transfers
  - – Striping (cf DPSS)
  - – Policy-based access control
  - – *Server-side computation*

    [later]

The slide from Steve Tuecke, ANL

And the Universal Protocol is … GSI-FTP

- Why FTP?
  - — enables interoperation with many commodity tools
  - — Already supports many desired features, easily extended to support others
  - — Well understood and supported
- We use the term “GSI-FTP” to refer to:
  - — Transfer protocol which meets our requirements
  - — Family of tools which implement the protocol
- Note GSI-FTP is much more than FTP
  - — despite the name, GSI-FTP is not restricted to file transfer!
  - — supports partial file read/write too

The slide from Steve Tuecke, ANL

GSI-FTP: Basic Approach

- FTP protocol is defined by several IETF RFCs
- Start with most commonly used subset
  - Standard FTP: get/put etc., 3rd-party transfer
- Implement standard but often unused features
  - GSS binding, extended directory listing, simple restart
- Extend in various ways, while preserving interoperability with existing servers
  - Striped/parallel data channels, partial file, automatic and manual TCP buffer setting, progress monitoring, extended restart

The slide from Steve Tuecke, ANL

Data Grids and Data Grid Performance Issues: Day 2

Brian L. Tierney (bltierney@lbl.gov)
Data Intensive Distributed Computing Group
Lawrence Berkeley National Laboratory
(currently on leave at CERN, IT/PDP/TE)

Outline

- Part I
  - — Introduction
  - — Data Grid Architecture
  - — Existing Data Grid Systems
    - DPSS and SRB
- Part II
  - — Current Data Grid Work
    - Globus + LBNL work
    - other data grid projects
- Part III
  - — Data Grid Performance Issues
    - network and TCP Issues
    - application design considerations

The GSI-FTP Family of Tools

- Patches to existing FTP code
  - — GSI-enabled versions of existing FTP client and server, for high-quality production code
- Custom-developed libraries
  - — Implement full GSI-FTP protocol, targeting custom use, high-performance
- Custom-developed tools
  - — Servers and clients with specialized functionality and performance

The slide from Steve Tuecke, ANL

Family of Tools: Patches to Existing Code

- Patches to standard FTP clients and servers
  - — gsi-ncftp: Widely used client
  - — gsi-wuftpd: Widely used server
  - — GSI modified HPSS pftpd
  - — GSI modified Unitree ftpd
- Provides high-quality, production ready, FTP clients and servers
- Integration with common mass storage systems
  - — these do not support the full gsi-ftp protocol

The slide from Steve Tuecke, ANL

Family of Tools: Custom Developed Programs

- Simple production client
  - — globus-url-copy: Simple URL-to-URL copy
- Experimental FTP servers
  - — Striped FTP server (i.e.: DPSS-2)
  - — Multi-threaded FTP server with parallel channels
  - — Firewall FTP proxy: Securely and efficiently allow transfers through firewalls
  - — Striped interface to parallel file systems
- Experimental FTP clients
  - — POSIX file interface

The slide from Steve Tuecke, ANLc

Current DPSS Work

- Turning the DPSS into a high-performance, parallel, striped, Grid-enabled FTP server
- new DPSS will support:
  - — multiple servers
  - — multiple data streams/server
  - — automatic network tuning
  - — GSI security
  - — advanced reservations
  - — globus-ftp clients
  - — will also work with all standard ftp clients

Replica Management

- Maintain a mapping between logical names for files and collections and one or more physical locations
- Important for many applications
  - — Example: CERN HLT data
    - Multiple petabytes of data per year
    - Copy of everything at CERN (Tier 0)
    - Subsets at national centers (Tier 1)
    - Smaller regional centers (Tier 2)
    - Individual researchers will have copies

The slide from Ann Chervenak, USC/ISI

Globus Approach to Replica Management

- replica cataloging and reliable replication are two fundamental Grid services
  - Layered on other Grid services: GSI, transport, information service
  - Use LDAP as catalog format and protocol, for consistency
  - Use as a building block for other tools
- Advantage
  - These services can be used in a wide variety of situations

This slide from Ann Chervenak, USC/ISI

Replica Manager Components

- Replica catalog definition
  - LDAP object classes for representing logical-to-physical mappings in an LDAP catalog
- Low-level replica catalog API
  - globus_replica_catalog library
  - Manipulates replica catalog: add, delete, etc.
- High-level reliable replication API
  - globus_replica_manager library
  - Combines calls to file transfer library (GSI-FTP) and calls to low-level replica catalogue API functions

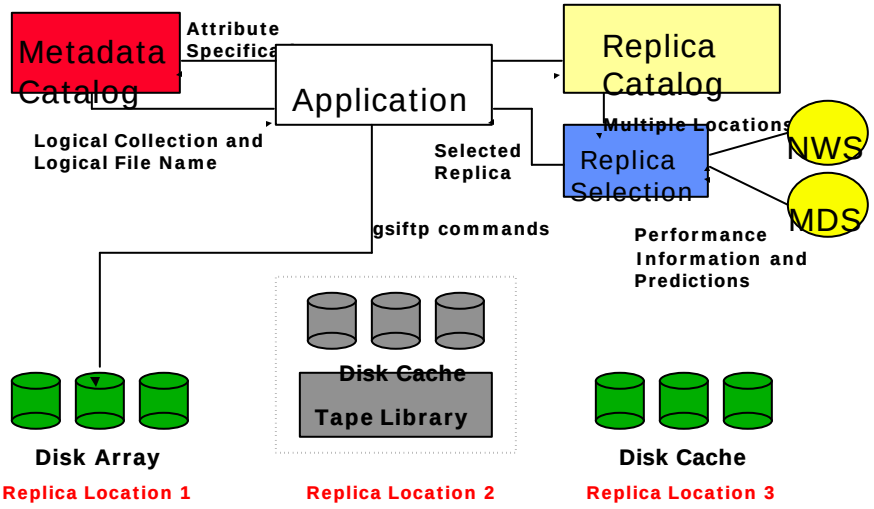
This slide from Ann Chervenak, USC/ISI

Replica Catalog Services as Building Blocks: Examples

- Combine with Globus Information Service to build replica selection services
  - e.g. “find best replica” using performance information from NWS and MDS
  - Use of LDAP as the common protocol for info and replica services makes this easier
- Combine with application managers to build data distribution services
  - e.g., build new replicas in response to frequent data access

This slide from Ann Chervenak, USC/ISI

A Model Architecture for Data Grids



This slide from Ann Chervenak, USC/ISI

Outstanding Issues for Replica Management

- Current architecture assumes a read-only workload
  - — What write consistency should we support?
- What high-level operations are needed?
  - — Combine storage and catalog operations
- Support replication in Objectivity DB?
- Replicating the replica catalog
- Replication of partial files
- Alternate catalog views: files belong to more than one logical collection

This slide from Ann Chervenak, USC/ISI

Other Data Grid Work

- Several active Data Grid Projects:
  - — IBP project
    - http://www.cs.utk.edu/~plank/IBP
  - — HEP Data Grid Projects
    - Partical Phyics Data Grid (PPDG)
      - — http://www.cacr.caltech.edu/ppdg/
    - GriPhyN Project:
      - — http://www.griphyn.org/
    - EU DATAGRID Project
      - — http://tilde-les.home.cern.ch/~les/grid
  - — Several others
    - The Grid Forum (http://www.gridforum.org) Remote I/O Working Grid has a paper describing several Data Grid applications
    - http://www.sdsc.edu/GridForum/RemoteData/Papers/

Internet Backplane Protocol (IBP)

- NSF funded project lead by University of Tennessee
  - http://icl.cs.utk.edu/ibp/
- Main ideas:
  - serves writable (not just readable) storage as a wide area network resource
  - provides 3rd party transfer facilities
  - decouples the notion of the user ID from storage
    - servers allow unauthenticated clients write access
    - files have a limited lifetime
- Currently deploying 6-10 IBP servers around the USA
- Developing "logistical ftp server":
  - allows ftp client to schedule a "get" sometime in the future, and the IBP ftp server will automatically move the file to the "closest" server to the client.

DATAGRID Project

- The overall topology is the MONARC Regional Center model
  - a number of national grids (equivalent to the Tier 1 and Tier 2 Regional Centers)
  - interconnected by a central node at CERN.
- Includes 12 "work packages"
  - WP2, "Grid Data Management", lead by CERN, is responsible for the issues we have been discussing today.
- For more information:
  - http://tilde-les.home.cern.ch/~les/grid

WP2: Data Management

- WP2 Objectives:
  - — secure access to massive amounts of data
  - — universal global name space
  - — move and replicate data at high speed from one geographical site to another
  - — manage synchronization of remote replicas
  - — automated wide-area data caching and distribution
    - based on dynamic usage patterns
  - — interfaces to heterogeneous mass storage management systems

DATAGRID and Globus

- DATAGRID project will be built using the Globus Toolkit:
  - — http://wwwinfo.cern.ch/pdp/te/globus/documentation.html
- You will learn about much more about this project and Globus next week!

End of Part II

Network Performance Issues

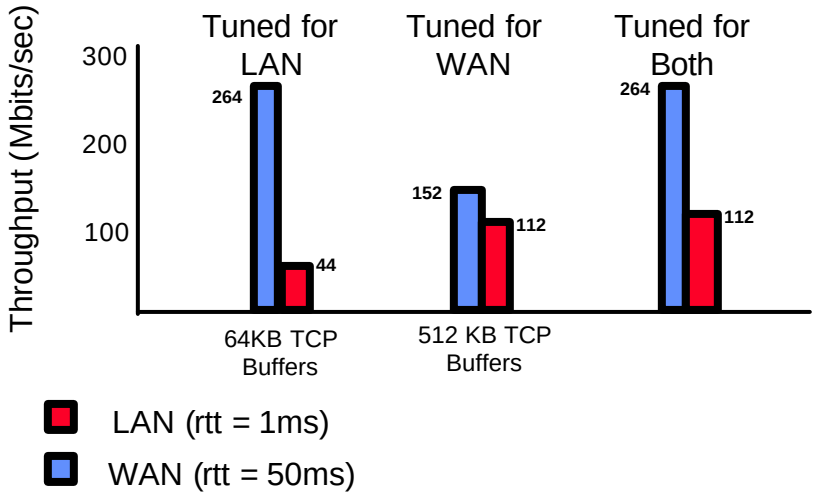
TCP Performance Tuning Issues

- Getting good TCP performance over high latency networks is hard!
- You must keep the pipe full, and the size of the pipe is directly related to the network latency
  - — Example: from LBNL to ANL (3000km), there is an OC12 network, and the one-way latency is 25ms
    - Bandwidth = 67 MB/sec (OC12 = 622 Mb/s = ATM and IP headers = 539 Mb/s for data
    - Need 67 MBytes * .025 sec = 1.7 MB of data “in flight” to fill the pipe
  - — Example: CERN to SLAC: latency = 84 ms, and bandwidth will soon be upgraded to OC3
    - assume end-to-end bandwidth of 12 MB/sec, need 1.008 MBytes to fill the pipe

Setting the TCP buffer sizes

- It is critical to use the optimal TCP send and receive socket buffer sizes for the link you are using.
  - — if too small, the TCP window will never fully open up
  - — if too large, the sender can overrun the receiver, and the TCP window will shut down
- Default TCP buffer sizes are way too small for this type of network
  - — default TCP send/receive buffers are typically 24 or 32 KB
  - — with 24 KB buffers, can only get 2.2% of the available bandwidth

Importance of TCP Tuning



TCP Buffer Tuning

- Must adjust buffer size in your applications:

```
int skt, int sndsize;
err = setsockopt(skt, SOL_SOCKET, SO_SNDBUF,
                 (char *)&sndsize,(int)sizeof(sndsize));
```

and/or

```
err = setsockopt(skt, SOL_SOCKET, SO_RCVBUF,
                 (char *)&sndsize,(int)sizeof(sndsize));
```

- Also need to adjust system max and default buffer
  - Example: in Linux, add to /etc/rc.d/rc.local

```
echo 8388608 > /proc/sys/net/core/wmem_max
echo 8388608 > /proc/sys/net/core/rmem_max
echo 65536 > /proc/sys/net/core/rmem_default
echo 65536 > /proc/sys/net/core/wmem_default
```

- For More Info, see: http://www-didc.lbl.gov/tcp-wan.html

Determining the Buffer Size

- The optimal buffer size is twice the bandwidth*delay product of the link:

  buffer size = 2 * bandwidth * delay

- The ping program can be used to get the delay
  - e.g.:
```
portnoy.lbl.gov(60)>ping -s lxplus.cern.ch
64 bytes from lxplus012.cern.ch: icmp_seq=0. time=169. ms
64 bytes from lxplus012.cern.ch: icmp_seq=1. time=169. ms
64 bytes from lxplus012.cern.ch: icmp_seq=2. time=166. ms
```
- pipechar or pchar can be used to get the bandwidth of the slowest hop in your path. (see next slides)
- Since ping gives the round trip time (RTT), this formula can be used instead of the previous one:

  buffer size = bandwidth * RTT

Buffer Size Example

- ping time = 50 ms
- slowest network segment = 10 MBytes/sec (e.g.: the end-to-end network consists of all 100 BT ethernet and OC3 (155 Mbps)
- TCP buffers should be:

  .05 sec * 10 = 500 KBytes.

- Remember: default buffer size is usually only 24KB, and default maximum buffer size is only 256KB !

pchar

- pchar is a reimplementation of the pathchar utility, written by Van Jacobson.
  - — http://www.employees.org/~bmah/Software/pchar/
  - — attempts to characterize the bandwidth, latency, and loss of links along an end-to-end path
- How it works:
  - — sends UDP packets of varying sizes and analyzes ICMP messages produced by intermediate routers along the path
  - — estimate the bandwidth and fixed round-trip delay along the path by measuring the response time for packets of different sizes

pchar details

- How it works (cont.)
  - — vary the TTL of the outgoing packets to get responses from different intermediate routers.
    - At each hop, pchar sends a number of packets of varying sizes
  - — attempt to isolate jitter caused by network queuing:
    - determine the minimum response times for each packet size
    - performs a simple linear regression fit to the minimum response times.
    - this fit yields the partial path bandwidth and round-trip time estimates.
  - — for per-hop estimates, pchar computes the differences in the linear regression parameter estimates for two adjacent partial-path datasets

pipechar

- Problems with pchar:
  - — takes a LONG time to run (typically about 40 minutes for an 8 hop path)
  - — often reports inaccurate results on high-speed (e.g.: > OC3) links.
- New tool called *pipechar*
  - — http://www-didc.lbl.gov/~jin/network/net-tools.html
  - — solves the problems with pchar, but only reports the bottleneck link accurately
    - takes about 2 minutes to measure a 8 hop path

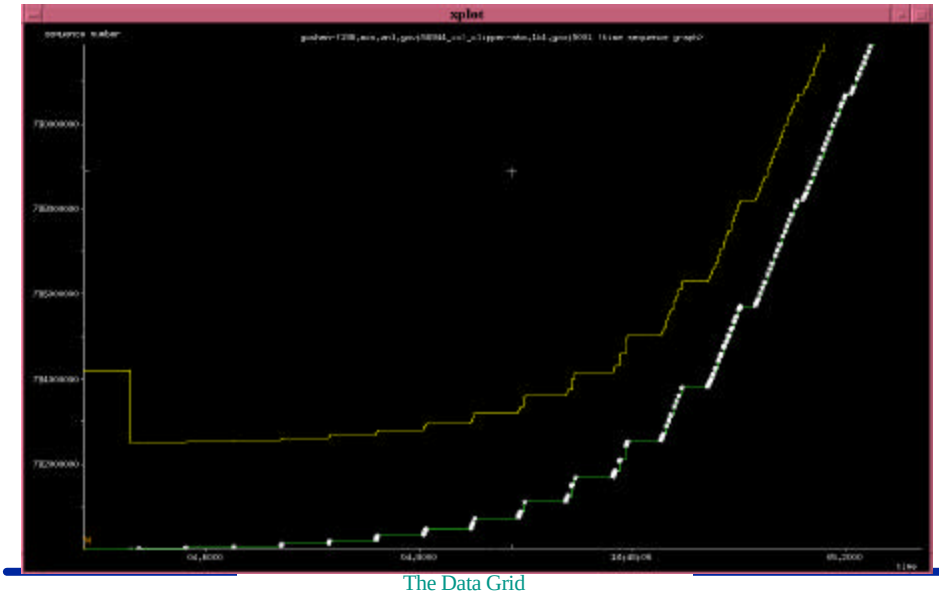
Sample pipechar output

```
>pipechar pdrd10.cern.ch
From localhost:  156.522 Mbps      (157.6028 Mbps)
1: ir100gw-r2.lbl.gov               (131.243.2.1 )
|       157.295 Mbps              <4.9587% BW used>
2: lbl2-gig-e.es.net                (198.129.224.2)
|       159.364 Mbps              <21.5560% BW used>
3: chicago1-atms.es.net             (134.55.24.17)
|       45.715 Mbps               <1.6378% BW used>
4:                                  (206.220.243.32)
|       46.895 Mbps               <1.6378% BW used>
5: cernh9-s5-0.cern.ch              (192.65.184.142)
|       46.330 Mbps               <5.9290% BW used>
6: cgate2.cern.ch                   (192.65.185.1)
|       45.348 Mbps               <10.6760% BW used>
7: cgate1-dmz.cern.ch               (192.65.184.65)
|       46.041 Mbps               <10.1195% BW used>
8: b513-b-rca86-1-gb0.cern.ch       (128.141.211.1)
|       45.411 Mbps    !!!       <23.0134% BW used>
9: b513-c-rca86-1-bb1.cern.ch       (194.12.131.6)
|       46.911 Mbps               <9.3956% BW used>
10: r31-s-rca20-1-gb7.cern.ch        (194.12.129.98)
|       9.954 Mbps *** static bottle-neck 10BT
11: pcrd10.cern.ch                    (137.138.29.237)
```

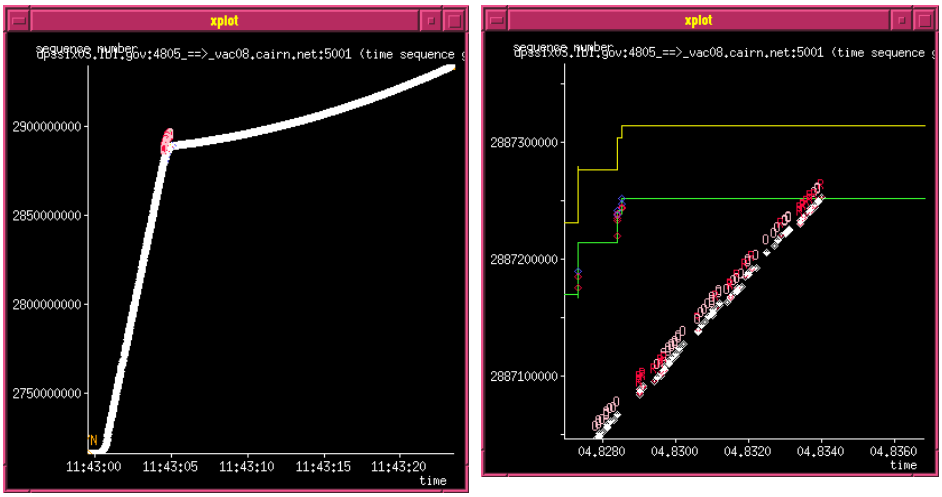
Other TCP Issues

- Other things to be aware of:
  - — TCP slow-start
    - On the LBNL to ANL link, it takes 12 RTT's to ramp up to full window size, so need to send about 10 MB of data before the TCP congestion window will fully open up.
  - — router buffer issues
  - — host issues

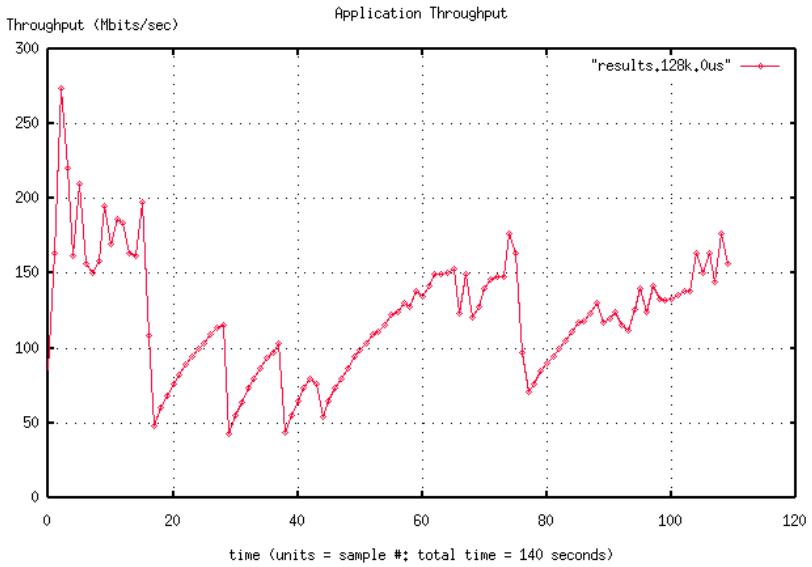
TCP Slow Start



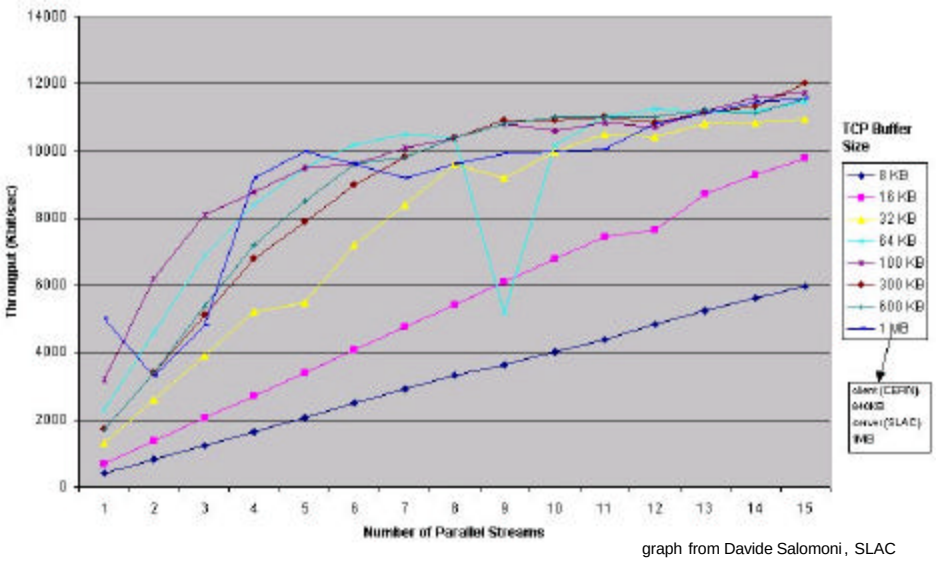
Problems with TCP over NGI-like Networks: Router buffer overflows



TCP Throughput on Uncongested DARPA SuperNet

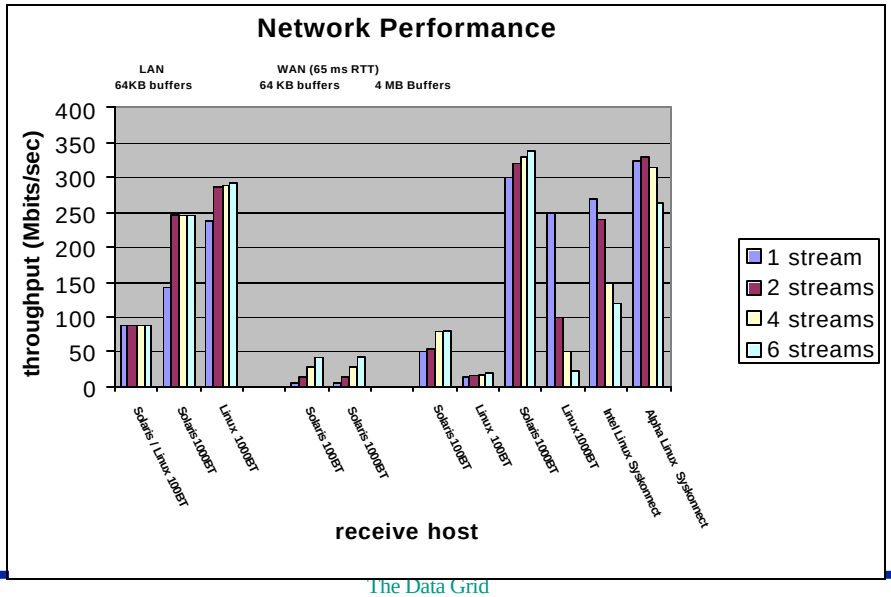


Advantage of Parallel Transfers



graph from Davide Salomoni, SLAC

TCP WAN Performance: Host Issues



Things to Notice in Previous Slide

- Parallel streams help a lot with un-tuned TCP buffers
  - — and help a little with large buffers on Solaris
- Problems sending from a 1000BT host to a 100BT Linux host
- Problems sending multiple streams to a 1000BT Linux system, especially with cheaper 1000BT hardware

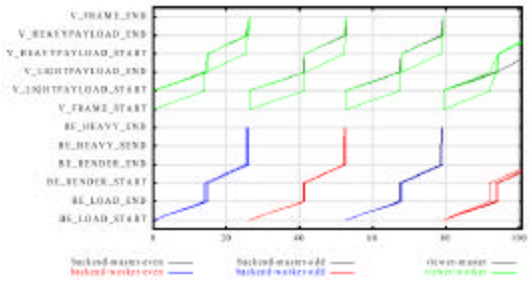
Application Design Considerations

Other Techniques to Achieve High Throughput over a WAN

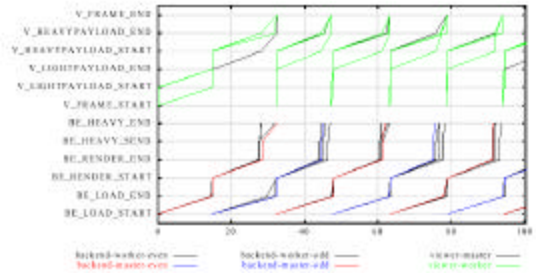
- Use multiple TCP sockets for the data stream
  - —if your receive host is fast enough
- Use a separate thread for each socket
- Keep the data pipeline full
  - —Use asynchronous I/O
    - overlap I/O and computation
  - —Read and write large amounts of data (> 1MB) at a time whenever possible
- Avoid unnecessary data copies
  - —manipulate pointers to data blocks instead

Use Asynchronous I/O

- I/O followed by processing



- overlapped I/O and processing



almost a 2:1 speedup

Throughput vs. Latency

- Most of the techniques we have discussed are designed to improve throughput
- Some of these might even increase latency
  - — with large TCP buffers, OS will buffer more data before sending it out.
- Goal of a Grid application programmer
  - — hide latency
- However, there are some ways to help latency:
  - — use separate control and data sockets
  - — use TCP_NODELAY option on control socket
    - But: combine control messages together into 1 larger message whenever possible on TCP_NODELAY sockets

For more Information

- For more information:
  - — http://www-didc.lbl.gov/
  - — email: bltierney@lbl.gov
- Other URLs:
  - — http://www.globus.org/
  - — http://www.gridforum.org/

References

- *"The Grid: Blueprint for a New Computing Infrastructure"*, edited by Ian Foster and Carl Kesselman. Morgan Kaufmann, Pub. August 1998. ISBN 1-55860-475-8.
- Schopf, J Nitzberg, B., " Grids: The Top Ten Questions", Grid Forum white paper. http://www.cs.nwu.edu/~jms/Pubs/
- Tierney, B. et. al., "A Data Intensive Distributed Computing Architecture for Grid Applications", Future Generation Computer Systems (an Elsevier Journal), vol 16, #5, April 2000.:
- Tierney, B. et. al "Using High-Speed WANs and Network Data Caches to Enable Remote and Distributed Visualization", accepted for publication in the Proceeding of IEEE Supercomputing 2000, Portland, OR, Nov 2000. (with LBNL Visualization Group)
- Gunter, D., "NetLogger: A Toolkit for Distributed System Performance Analysis", ACM/IEEE MASCOTS 2000, Eighth International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems, Aug, 2000.
- Tierney, B. et. al "Using NetLogger for Distributed Systems Performance Analysis of the BaBar Data Analysis System" , Proceedings of Computers in High Energy Physics 2000 (CHEP 2000) , Feb. 2000.